U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
OFFICE OF SCIENCE AND TECHNOLOGY
METEOROLOGICAL DEVELOPMENT LABORATORY

MDL SOFTWARE DEVELOPMENT PROCESS (SDP) FOR THE AWIPS PROJECT

DRAFT
February 13, 2004

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Section **Page**

Figures

Tables

MDL SOFTWARE DEVELOPMENT PROCESS FOR THE AWIPS PROJECT

1.0 INTRODUCTION

The Meteorological Development Laboratory (MDL) Software Development Process (SDP) for the Advanced Weather Interactive Processing System (AWIPS) project establishes the software and development processes that are used throughout the MDL software development life cycle. This document describes the processes and procedures that are used to design, implement, and test MDL software.

1.1 DOCUMENT ORGANIZATION

The SDP is organized into the following ten sections.

Section 1 - The Introduction presents the purpose and scope of the plan, an overview of the project, and related documentation.

Section 2 - The Project Organization and Responsibilities section discusses the project organization and the roles and responsibilities for MDL and other organizations that interface with MDL during the development process.

Section 3 - The Deployment Strategy section describes the development and release strategy for software developed by MDL under this SDP.

Section 4 - The Software Development Process section presents an overview of the software development life cycle and describes the major activities in each life cycle phase. It establishes the software development process, methods, and standards to be used in the development of AWIPS applications.

Section 5 - Documentation describes documentation created and the method for the retention of that documentation.

Section 6 - Reviews describes the types of reviews, what is being reviewed, when reviews are employed, and policies and procedures associated with each review

Section 7 - Testing explains the concept and activities employed in testing AWIPS applications.

Section 8 - The Quality Assurance section presents concepts and activities used to ensure a quality software product.

Section 9 - The Configuration Management section describes the concepts and activities used for the management of the software development and test products.

Section 10 - Software Standards identifies the standards pertaining to software development including coding standards and design rules.

Section 11- The Environment section describes the tools, hardware and software environments used to develop and test AWIPS applications.

1.2 SUPPORTING DOCUMENTS

Other important documents related to the SDP include:

Configuration Management Plan for AWIPS (NWS 2003a) - This plan describes the detailed configuration management strategy for applying version control and change management to MDL-developed AWIPS software. It defines the roles, responsibilities, tools and techniques used for managing software during the development life cycle. In addition, it describes the configuration management strategies for managing documentation, support software and hardware used within the AWIPS development environment.

MDL Standards, Guidelines and Procedures (NWS 2003b) - This document contains the software standards used in the development of AWIPS applications and contains the set of procedures and guidelines to be used by the development team to standardize the development process and ensure that it is a repeatable process.

Installation & Testing Process on the AWIPS Test Systems: NMTW, NMTR, and NHDA (NWS 2002e) - This document describes three AWIPS test systems that are on the AWIPS WAN and describes the process of installation and testing on these systems. For each system, there is a description of their purpose, capabilities, restrictions, external interfaces, test interfaces, and date sources. Procedures for the use of these test systems and restrictions on their use are also provided. In addition, this document describes the two Office of Science and technology (OST) software development and build suites of computer systems and the process involved in moving software between systems.

2.0 PROJECT ORGANIZATION AND RESPONSIBILITIES

This section describes the responsibilities within MDL, and the Government and Contractor organizations that interface with MDL during the development of application software.

2.1 AWIPS PROJECT DEFINITION OVERVIEW

Interactive Forecast Preparation System (IFPS) - With the IFPS, forecasters prepare graphical depictions of predicted weather, using interactive displays of initial forecasts. With tools like the Forecast Systems Laboratory (FSL) Graphical Forecast Editor Suite (GFESuite), forecasters can initialize and edit grids of forecast weather elements (e.g., maximum and minimum temperatures, wind speed and direction, and amount of cloud cover). These grids make up a common digital database which is used by product generation tools to automatically compose and format a suite of routine public, marine and fire weather forecast products for the National Oceanic and Atmospheric Administration (NOAA) Weather Wire Service (NWWS) and NOAA Weather Radio (NWR).

Aviation Forecast Preparation and Monitoring - Tools for interactive composition, editing, and quality control of the forecaster-issued TAF product, and automated monitoring.

Climate - Product formatters which produce a daily, F6, monthly, seasonal and yearly climate reports for the NWWS and the NWR.

Hourly Weather Report - The Hourly Weather Roundup (HWR) program is a summary of the current weather observations for locations in and around the local area, including marine observations for those Weather Forecast Offices (WFO's) with coastal responsibilities. This product is produced for NWWS and the NWR formatted products.

Verification - The AWIPS Verification Program is the replacement for the AFOS-Era Verification (AEV) Program. It allows the NWS to maintain an uninterrupted set of historical, national forecast verification statistics from the pre-AFOS era through the AWIPS transition.

System for Convection Analysis and Nowcasting (SCAN) - Convection is a mechanism for heat transfer. In the atmosphere, convection can produce rising air currents and thunderstorms. SCAN is a sophisticated, state-of-the-art software package. SCAN detects, analyses, and monitors thunderstorms and generates short-term forecasts and warning information for severe and tornadic thunderstorms and flash floods. SCAN is a tool to help warning forecasters make better decisions.

Other Decision Assistance Tools - Activities to implement a suite of decision assistance tools for severe weather, marine, aviation, winter weather, and fire weather. These tools include the System for AWIPS Forecasting and Evaluation for Seas and Lakes (SAFESEAS), the System for Nowcasting of Winter Weather (SNOW), the Aviation Weather Monitor (Airport-Wx), the Fire Weather Monitor (Fire-Wx) and the General User Alert Display Panel (GUARDIAN).

Watch, Warning and Advisory (WWA) Applications - The WWA application enables the user to create, modify, and maintain watch warning and advisory products. After creation, the forecaster may follow up, clear, or cancel the statement within the WWA application. The WWA application within IFPS is used for long-fused hazards. In addition, any product produced using WWA is formatted for transmission to NOAA Weather Radio (NWR).

Local Storm Reports (LSR) - Product formatter that prepares a Local Storm Report for NWWS.

Flash Flood Monitoring and Prediction (FFMP) - FFMP provides a framework to evaluate and monitor quantitative precipitation estimates (QPE's) within the context of the flash flood problem.

Radar Mosaics Operations and Development - maintains and refines operational radar mosaic products and is developing new mosaic products at substantially higher spatial and temporal resolution.

2.2 OTHER ORGANIZATIONS AND CONTRACTORS

MDL staff must interface regularly with other organizations that play a critical role in the development and deployment of AWIPS software. Weekly AWIPS Conference calls are conducted to share information and to coordinate the technical exchange between organizations.

RS Information Systems (RSIS)/Science Applications International Corporation (SAIC) - RSIS is responsible for managing and staffing MDL's configuration management, quality assurance and system administration groups and for providing software developers who participate in AWIPS development.

Northrop Grumman Information Technology (NGIT) - NGIT is responsible for design, developing, documenting and maintaining the network segment of AWIPS, supporting the integration and delivery of AWIPS software, and providing task support of other government development activities as needed. Also, NGIT provides contract support to MDL in support of the development of hydromet applications.

Office of Science and Technology (OST) - OST has overall management responsibility for the AWIPS program.

OST/Software Engineering Center (SEC) - SEC has overall responsibility for the AWIPS development effort, managing the distributed development across all development organizations. They have oversight over a number of technical teams such as the Software Engineering Group, Systems Engineering Team, Communications Working Group, Hardware Working Group and Performance Working Group. The SEC is responsible for reviewing the software plans for appropriateness and completeness, especially in relationship to the overall engineering plans. The SEC Systems Engineer deals with system performance issues and has authority and responsibility for making the system meet performance requirements.

OST/Program and Plans Division (PPD) - PPD manages execution of program management and development programs. PPD reviews science and technology options, prepares solutions and develops plans to meet service requirements. PPD plans, coordinates, and manages a technical infusion and evolution program. PPD identifies and monitors objective measures of program progress.

Office of Climate, Water and Weather Services (OCWWS) - OCWWS represents the user community for the development of AWIPS applications. They determine and validate WFO user requirements, assure user requirements and feedback are incorporated into the development cycle, conduct evaluation of services and identify enhancements to be incorporated into AWIPS.

Forecast Systems Laboratory (FSL) - FSL is responsible for the development of AWIPS site system software, development of selected applications including WFO Advanced (WFO-A) Hydromet applications and the Graphical Forecaster Editor (GFE Suite).

Hydrologic Research Laboratory (HRL) - HRL is responsible for the development of hydrologic application software for WFOs and River Forecast centers (RFCs).

Office of Operational Systems (OOS) - OOS provides change management, production software configuration management in support of AWIPS, and manages the Site Support Team (SST), the programs point contact for the field offices.

Chief Information Officer (CIO) - CIO manages the Network Control Facility (NCF) which monitors network and site performance and supports users in troubleshooting problems.

3.0 PROJECT MANAGEMENT

This section describes a standard approach and mechanism for project managers to plan, track, and measure the software development process.

3.1 RESPONSIBILITIES

The responsibilities of project management are described below.

Director, MDL - Overall responsibility for the software development effort.

Project Manager (PM) - Usually the Project Manager is the MDL branch chief responsible for the development of the application. The Project Manager is responsible for reviewing the plans, making the project commitments, and reviewing any changes. Specifically, the Project Manager oversees cost, schedule, and interfaces with other NWS organizations. The Project Manager conducts regular oversight reviews with the MDL Director and MDL Deputy Director. The Project Manager assigns the Task Manager.

Task Manager - The Task Manager's responsibilities include the assignment of Task Leads, and verifying that the resources and completion dates have been confirmed. Task Manager will call a Kick Off meeting to pass this information to the Task Leads and to the assigned Developers. It is also the Task Manager's responsibility to assign, if necessary, a Design Lead and a Testing Coordinator. The Task Manager also communicates with Configuration Management and System Administration to establish work spaces, equipment requirements and version verification of all supporting tools for development and testing. Both parties are kept in the loop for any changes in requirements or needs. The Task Manager reports to the Project Manager.

3.2 MANAGEMENT PROCESS

Managing a software project requires careful planning, control of activities, and tracking against the planned activities. Once a plan is developed, the actual activities are tracked against the plan to determine whether there is any deviation from the plan. Metrics (as defined in Section 4.2) are used to measure performance and suggest process improvement.

This management process will mature as the program progresses. This means that the plans are living documents. This management process emphasizes early planning and risk analysis. The plans are reviewed, revised, and expanded based on the most recent knowledge of the program at key milestones, when the scope changes, and at regular intervals.

3.2.1 PLANNING

Software project planning involves developing estimates for the work to be done, establishing the necessary commitments, and defining a plan to complete the work. The plan provides the basis for initiating the software effort and managing the work. Accurate estimations of cost and schedule up front and adherence to required staffing levels and equipment usage are a key factor to completing a project within budget.

Project planning is a continuous process. As the development proceeds, certain design decisions may change the plan or schedule. As change requests (e.g., new deficiencies, enhancements, or requirements) are identified, the process must be repeated and documentation produced describing the impact of any changes to the cost and schedule.

The following functions are required during the Planning step. The responsible party for each function is shown in **bold** type.

Create AWIPS Change Request (ACR) - An ACR starts the AWIPS development cycle. ACRs can be initiated by **OS**, **SEC**, and **PPD** and are prepared for new software requirements, software enhancements, or hardware and system changes.

Create Task Development Report (TDRs) - The **Task Manager** establishes a set of TDRs based on the known ACRs.

Create Software Problem Reports (SPRs) - For software deficiencies the **Task Manager** establishes a SPR for each reported software deficiency. Software deficiencies are reported as Trouble Tickets (TTs) or were created based on the testing of a previous software release.

Estimate Level of Effort (ELOE) - For each TDR and SPR the **Task Manager** estimates the size of the task and the ELOE. The size of the task includes the number of software components required and lines of code. The ELOE is measured by estimating the number of labor hours required to develop, test, document, and maintain the TDR or SPR.

Assign Target Release - The **Task Manager** assigns the TDRs and SPRs to a Target Release.

Assign Task Lead - The **Task Manager** assigns the TDRs and SPRs to a Task Lead.

Prepare Project Tracking Information (PTI) - The **Task Manager** prepares a Staffing Plan and Development Schedule.

Approve Staffing Plan - The **Software Manager** submits the Staffing Plan to SEC. **SEC** approves the Staffing Plan.

Plan Development and Test Environment - The **Task Manager** works with the **CM** to determine work set to be used, how and when it will be populated, and the build environment and schedule needed. The **Task Manager** notifies the **SAM** of the test environment required and the schedule to be met.

Initiate Development - In accordance with the schedule, the **Task Manager** actions the appropriate TDRs/SPRs to begin the development cycle.

Table 3.2.1-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Planning step along with the responsible parties.

Table 3.2.1-1. Planning Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	AWIPS Change Requests (ACRs) prepared by MDL	OCWWS, PPD, SEC
Input	ACR received	OCWWS, PPD, SEC
Output	Project Tracking Information (PTI)	Task Manager
	TDRs/SPRs assigned a target release and Task Lead	Task Manager
Exit Criteria	Staffing Plan is updated and approved by SEC	SEC
	TDRs/SPRs in Task Leads pending list.	Task Manager
	Project Tracking Plan completed	Task Manager
	Approved Staffing Plan is baselined	CM
	CM environment and schedule determined	Task Manager/CM
	Test environment and schedule determined	Task Manager/SAM
Process Controls	SQA process audits	SM
	Peer reviews	Task Manager
	Metric collection, analysis and reporting	SM
Metrics	Compare estimated to actual Level of Effort (LOE)	SM

3.2.2 PROJECT TRACKING AND OVERSIGHT

Software project tracking and oversight involve tracking and reviewing software progress against the documented estimates, schedules, and plans, and adjusting these based upon the actual data. This activity occurs throughout the development effort.

Software project tracking and oversight start as soon as the effort commences. The Staffing Plans and Development Schedules are used as the basis for tracking and oversight throughout the project. During development, actual data are collected according to the process model. These data are analyzed at specified intervals against the plan. If the actual status of the program deviates beyond an acceptable norm, corrective action is taken. Corrections made to schedule, cost, or software sizes are reviewed with respect to each other. Then, the plans and any renegotiated commitments are revised and reviewed.

When modifying the schedule, the Software Manager keeps records that explain the reasons for various corrective actions and the rationale for that change. This information is useful for developing the lessons learned and other postmortem analysis.

The following are reviews that are established to track development projects.

Status Reviews - Status reviews are conducted with the Task Manager, Task Lead and developers to review progress and address issues. These can be conducted weekly or biweekly depending on the application and point in the development process.

Planned Reviews - Planned reviews are conducted at key periods during the development cycle to review requirements, design, coding, testing, and delivery. In addition, reviews are conducted to review code (e.g., code walkthrough) and other development products (e.g., peer reviews). See Reviews (Section 6.0) for more information on these reviews.

Coordination Meetings - The Hydromet Engineering Design Group (HEDGe) meeting is conducted weekly to coordinate between Project Managers, Task Managers, Task Leads, CM and SAMs.

AWIPS Conference Calls - Weekly meetings to coordinate activities among other NWS organizations. The Software Manager attends these meetings.

MDL Staff Meeting - The MDL staff meeting is conducted weekly to coordinate between the MDL Director, Branch Chiefs and Software Manager.

Working Group Meetings - Monthly meetings with working groups consisting of developers and users of an application (e.g., IFPS, WWA) to review progress, prioritize requirements, and solicit feedback from the user community.

In addition, following information is used by the Task Manager to provide project tracking and oversight, and facilitate reporting to other NWS organizations.

Project Tracking Information (PTI)

- Development Schedule - High level schedule showing the development activities (e.g., requirement review) and major checkpoints on a time line.

- Staffing Plan - List of TDRs/STDRs per release with estimated level of effort and identified Task Manager, Task Lead and development staff.

- Development Checklist - Internal detailed checklist employed by the management to track development progress and adherence to the MDL Software Development Approach requirements (NWS 2004a).

See the MDL Standards, Guidelines, and Procedures (NWS2003a) for a PTI template.

Metrics - Information to guide process improvement (See Section 4.2).

Audits - Verification that the SDP is being followed.

4.0 SOFTWARE DEVELOPMENT PROCESS

This section describes the software development and maintenance process in detail. Below are the major activities that should be accomplished during the development of applications.

- Requirements Analysis
- Design
- Coding and Unit Testing
- Build and Release
- Software Integration Testing
- System Integration Testing
- Software Delivery

Some of these activities may not be necessary for specific types of projects. The process is flexible and can be ordered to fit any of the popular paradigms of software architecture including Waterfall, Rapid Prototype, or Evolutionary models. Figure 4.0-1 shows the typical waterfall software development process sequence with interrelationships, and significant checkpoints and reviews.

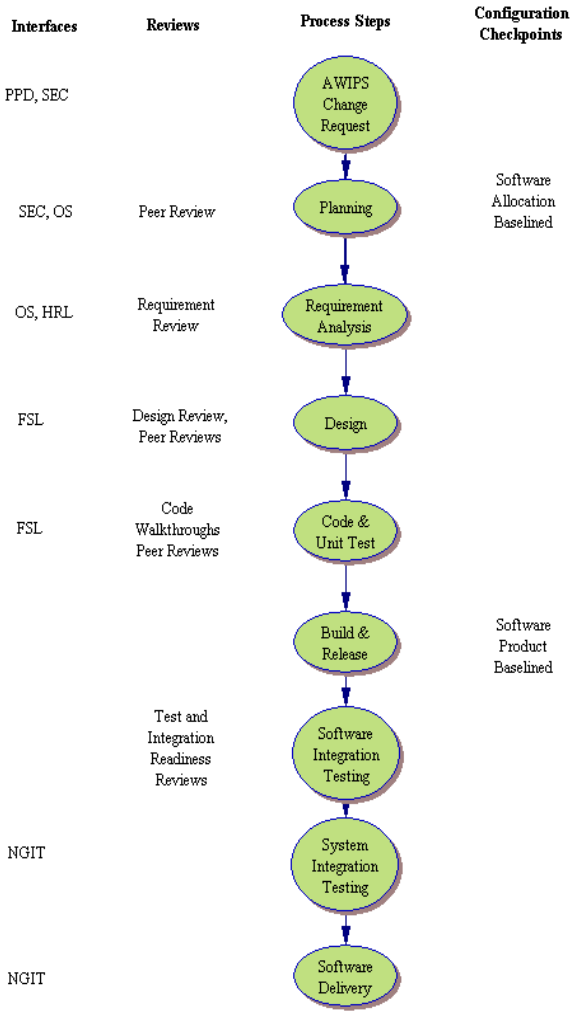

Figure 4.0-1: Data flow of the Software Development Process.

4.1 RESPONSIBILITIES

The development organizational definitions for each role along with their specific responsibilities are described below and in Figure 4.1-1. Depending on the size of the development effort, a staff member may be tasked with multiple roles.

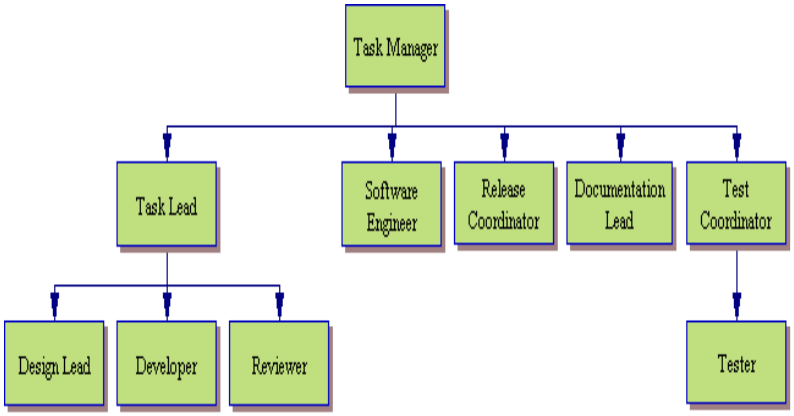

Figure 4.1-1. Development Organization

Design, Development and Testing (DDT) Teams - DDT teams perform the actual software development of the WFO HM applications. These teams consist of two to five developers and are made up of personnel from MDL and support contractor staff. The DDT team may include a Task Lead, Design Lead, Developers, Reviewers, and Testers.

Task Lead - The Task Lead leads the development of the TDRs/STDRs/SPRs. The Task Lead coordinate the activities of the Design Lead, Developers, and reviewers. The Task Lead performs the requirement analysis, assists with the design, reviews developers code and test procedures, and assists the Test Coordinator prepare test plan and schedules. The Task Lead reports to the Task Manager.

Software Engineer - The Software Engineer is the chief architect and leads the requirement analysis peer reviews and code walkthroughs. The Software Engineer reports to the Task Manager.

Release Coordinator - The Release Coordinator coordinates the software release with the Task Leads contributing software to that release, and prepares the software and documentation for delivery. The Release Coordinator reports to the Task Manager.

Design Lead - The Design Lead's responsibility is to create detail Design Documents for the individual tasks. The Design Lead conducts design review meetings so that design changes can be communicated to appropriate parties. The Design Lead reports to the Task Manager.

Developer - The developer's major responsibility is to code software using the established coding standards, guidelines and procedures. After completing the coding phase, the developers' go through the code review process with their Task Leads. It is also the developer's responsibility to create test procedures and to perform the unit testing.

Test Coordinator - The test coordinator's responsibility includes preparing a test matrix and test schedule and ensuring that the test coverage is complete. If it is not complete, the test coordinator either will create additional test cases or will ask for the developers help to create the specific test cases. The Test Coordinator will report the test results and total test coverage to the Task Manager on an establish schedule.

Tester - A developer, assigned by the Task Manager, preferably someone other than the developer who wrote the code.

Reviewers - The Reviewers role is to participate in the various reviews described in Section 6.0 of this document. The reviewers can be technical staff members, developers, specialists, or users whose backgrounds give them insight into the material to be discussed.

Documentation Lead - The Documentation Lead is responsible for coordinating the preparation and review of documentation for the application.

4.2 METRICS

The Project/Software Managers use software metrics gathered to evaluate key characteristics of the software being developed, the process employed, and the associated management indicators of progress. A successful metrics program depends on accurate and consistent data collection and presentation. The validity of the data should be determined prior to any analysis activity. Under these circumstances, the metrics can provide early warning of potential software development problems. In turn, this should lead to early problem resolution.

Graphic presentation of the metrics can reveal developing trends which, when analyzed as related sets, highlight anomalies that might otherwise be overlooked. Management can then determine their significance and corrective action can be taken. Results are used to improve the ongoing project and are reported at management reviews.

Types of metrics include number of software requirements, number of software requirement changes, product size, level of effort, cost, schedule, defects and computer resource utilization. In the future, MDL will develop composite metrics to indicate productivity, quality, production rate, and stability.

Metrics used by MDL are identified in Section 4.3 of this document.

4.3 DETAILED SOFTWARE PROCESS ACTIVITIES

For each step in the process, this document defines:

Entrance Criteria - criteria needed to start the activity,
Input - products dependencies of the step,
Functions - process and tasks of each step,
Metrics - set of measurement data resulting from the work products,
Responsibility - who is responsible for completing that activity,
Output - products of activity,
Exit criteria - criteria for completing the activity, and
Process controls - controls put into place to insure quality.

Also, any guidelines and process descriptions necessary to complete the activity will be referenced and kept in the MDL Standards, Guidelines and Procedures (NWS 2003b).

4.3.1 REQUIREMENTS ANALYSIS

The software development processes are requirements driven. Requirements are a formal statement of an attribute to be possessed by the product or a function to be performed by the product. These requirements form a written agreement between developer and customer. Requirement analysis provides a means for establishing and maintaining requirements so that both the customer and the developers are working from the same set of expectations.

The completion of the Project Planning (Section 3.2.1) activity results in a group of TDRs being assigned to a release and a Task Lead. The Task Lead defines and develops software requirements and prepares a Requirements Description (RD). The RD is a set of detailed software requirements derived from the AWIPS Change Request (ARC) linked to the TDRs. Requirements can address operation concepts, functional and user interface specifications, performance and capability, external interfaces, security, error handling, installation, configuration, language, maintenance, and the use of legacy software. For data-driven and data-intensive systems, the RD can include data sources, types, and rates. A compliance matrix is used to map software and derived requirements to TDRs.

A Requirement Review is held to finalize the understanding of requirements with the customer and obtain approval to develop the application or enhancement. Approval of the RD and Requirement Review material is provided by the appropriate approval body. For AWIPS, OCWWS is the approving body.

Note that requirements can be dynamic and change as the design is evaluated and development is conducted. As these changes occur, the changes should be made to the RD and those changes presented at the appropriate review.

The following functions are required during the Requirements Analysis step. The responsible party for each function is shown in **bold** type.

Define and develop software requirements - The **Task Lead** defines the set of software and derived requirements for this project.

Analyze requirements - A peer review is conducted by the **Task Lead** to analyze the requirements to ensure, at a minimum, traceability, completeness, clarity, testability, safety, and validity.

Schedule Requirements Review (RR) - A RR is scheduled by the **Task Lead** to present the requirements to the user community. A Requirement Review presentation is prepared and distributed to the participant at least one week prior to the review.

Perform Requirements Review (RR) - A RR is conducted by the **Task Lead.** Participants should include Project Manager, Task Manager, Regional Focal Points, Software Engineer, Software Manager, Configuration Management, DDT team and the approval body (e.g., OCWWS representatives for AWIPS development). Action items (if necessary) are documented and delivered to the approval body and tracked through closure by the **Task Lead.**

Approval - The RR is formally approved by **OCWWS** and **SEC.** The approval can be provided in an email, or memo.

Prepare Detailed Development Schedule - The **Task Lead** prepares an internal detailed schedule employed by the Task Manager to track development progress.

Update Project Tracking Information (PTI) - The **Task Manager** updates the Development Schedule and Staffing Plan with any changes that result from a better understanding of the requirements. The **Task Manager** will communicate these changes to the Software Manager.

Table 4.3.1-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Requirements Analysis step along with the responsible parties.

Table 4.3.1-1. Requirements Analysis Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Staffing Plan is approved by SEC	Task Manager
Input	ACRs, FRs	OCWWS, PPD, SEC
	Project tracking Information (PTI)	Task Manager
Output	Requirement Description	Task Lead
	Requirement Review documentation	Task Lead
	Updated Project Tracking Information	Task Manager
Exit Criteria	Requirements Description is approved	OCWWS
Process Controls	Requirements Review	Task Lead
	SQA process audits	SM
	Peer Reviews of Requirement Description	Task Lead
	Metrics, collection, analysis and reporting	SM
Metrics	To Be Determined (TBD)	

4.3.2 DESIGN

The Design phase will establish a complete software design to be used by the developers of the application. Use cases are created to further define the requirements. The software components are defined in terms of purpose, use cases, interfaces, data requirements, data structure, error handling, storage and throughput, timing requirements, and diagnostic considerations. The relationship of components is defined in terms of data flow between them and external interfaces, and the control flow between components.

A Design Review is held to finalize the understanding of design with the system engineering community and obtain approval to code the application or enhancement. Approval of the design and Design Review is provided by the appropriate approval body.

In most cases only one Designing step is required, however an optional step on Detail Design Process and its Review can be added for complex functionality.

The following functions are required during the Design step. The responsible party for each function is shown in **bold** type.

Partition software - The **Design Lead** defines the set of software components. The Design Lead establishes a set of Sub-Task Development Reports (STDRs) related to a TDR.

Prepare Use Cases - The **Design Lead** prepares use cases.

Prepare design - The **Design Lead** defines uses cases, component interfaces, relationships and data flows, physical structure, user interface, data structure, and critical test scenarios.

Peer Review - Peer reviews are conducted by the **Design Lead** to review selected design components. Participants could include the Task Manager, Task Lead and DDT team.

Schedule Design Review - A design review is scheduled by the **Task Lead** to present the design to the user representatives, system engineering and integrator community. A Design Review presentation is prepared and distributed to the participant at least one week prior to the review.

Update DAR Checklist - The **Task Lead** responds to the appropriate items in the DAR Checklist.

Perform Design Review - A design review is conducted by the **Task Lead**. Participants should include Project Manager, Task Manager, Software Engineer, Software Manager, Quality Assurance, Configuration Management, DDT team and the approval body (e.g., SEC representatives for AWIPS development). Action items (if necessary) are documented and delivered to the approval body and tracked through closure by the **Task Lead**. Note: SEC requests that Design Approach Reviews (DARs) be performed for certain applications. The above Design Review can substitute for a DAR.

Approval - The design review is formally approved by the **Task Manager**. The approval can be provided in an email, or memo.

Assign developers - Based on the Staffing Plan, the **Task Lead** assigns developers to the STDRs/SPRs and actions them to "Development."

Update Project Tracking Information (PTI) - The **Task Manager** updates the Staffing Plan and Development Schedule.

Table 4.3.2-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Design step along with the responsible parties.

Table 4.3.2-1. Design Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Requirements Description is approved	OCWWS
Input	Requirement Description	Task Lead
	Requirement Review documentation	Task Lead
	TDRs/SPRs	Task Manager
Output	Design Document	Design Lead
	Design Review Document	Design Lead
	Design review action items	Task Lead
	Use cases	Design Lead
	STDRs/SPRs	Task Lead, Task Manager
	DAR Checklist (partially completed)	Task Lead
Exit Criteria	Completion of design documentation	Design Lead
	Peer review conducted	Design Lead
	Design review conducted and approved	Task Lead, OCWWS, SEC
	STDRs assigned to the developer(s)	Task Lead
Process Control	Design Review	Design Lead
	SQA process audits	SM
	Peer Reviews of design documents	Design Lead
	Peer review of use cases	Design Lead
	Metrics, collection, analysis and reporting	Task Lead
Metrics	TBD	

4.3.3 CODE AND UNIT TESTING

Once the Design Review is approved and the Design Documents have been completed, the developers assigned to the Coding activity can begin.

Coding is performed uniformly across software products using defined standards and guidelines. See the MDL Standards, Guidelines and Procedures (NWS 2003b) for the applicable standards and guidelines. The objectives of source code written are as follows:

- Meets requirements
- Contains correct logic and interfaces and handles data structures properly as specified in the design documentation
- Complies with coding standards
- Compiles successfully, without any warning or error messages
- Follows good coding techniques
- Includes proper internal documentation
- Follows reasonable and understandable size limits
- Considers reuse, portability, and system independence

The following functions are required during the Code and Unit Testing step. The responsible party for each function is shown in **bold** type.

Prepare code - The code is created or modified by the **Developer** according to the design documents and MDL Standards, Guidelines and Procedures (NWS 2003b)

Create Test Procedures - Test procedures are created by the **Developer** to test the modification to the code. Additional test procedures are identified to ensure the modification has not affected the existing code. See the MDL Standards, Guidelines and Procedures (NWS 2003b) for more details on how to prepare test procedures.

Test Procedure Peer Review - The **Developer** should conduct a peer review of the test procedures.

Perform Unit Testing - The **Developer** should perform unit testing in accordance with the MDL Standards, Guidelines and Procedures (NWS 2003b).

Perform Code Walkthroughs - Once the code has been completed and Unit testing has been completed, the **Developer** prepares and performs a Code Walkthrough. This is typically an informal process, however, a Code Walkthrough form needs to be filled out and defects recorded.

Fix Defects - Once defects found during the Code Walkthrough are fixed by the **Developer** and approved by the **Reviewers**, the **Developer** can check the code in for Task Lead's Review.

Software Review - The **Task Lead** reviews the code and test procedures and works with the Developer to address any issues.

Prepare Installation Instructions - The **Release Coordinator** will create/modify install instruction. The **Release Coordinator** should initiate a peer review for the installation instructions.

Update DAR Checklist - The **Task Lead/Developer** responds to the appropriate items in the DAR Checklist.

Arrange Build schedule with CM - A build schedule for testing is coordinated between **CM**, and the **Task Lead** and **Task Manager.**

Conduct a Test Readiness Review (TRR) - The **Task Lead** will conduct a review of the coding activities and determine the readiness for Software Integration Testing (SwIT). Participants should include Project Manager, Task Manager, Task Lead, Software Manager, Configuration Management, and DDT team. Action items (if necessary) are documented.

Task Lead Approval - If the software is approved at the TRR, the **Task Lead** will action the STDRs to "approved."

Notify CM - The **Task Lead** will notify CM that the software is ready for build, release and testing.

Table 4.3.3-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Code and Unit Testing step along with the responsible parties.

Table 4.3.3-1. Code and Unit Testing Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Approved design documents	Design Lead
	Change documents (SPR/STDRs) delegated to Developer	Task Lead, Task Manager
Input	Design Documents	Design Lead
	Use Cases	Design Lead
	STDRs/SPRs under work	Task Lead, Task Manager
Output	Checked-in code	Developer
	Completed Code Walk through form	Developer
	New or updated Test Procedures	Developer
	Installation Instructions	Release Coordinator
	List of Test Procedures for integration test-ing	Developer
	Reviewed STDRs/SPRs in CM's pending list	Task Lead, Task Manager
	Updated DAR Checklist	Task Lead
Exit Criteria	Completion of Unit Testing	Developer
	Completion of the Code Walkthrough	Developer
	Completion of Test Procedures	Task Lead
	Approved status on STDR/SPRs	Task Lead
	Completion of Peer Reviews	Task Lead
	Delegate Tester for STDR/SPR	Task Lead
	Successful nightly development builds	Task Lead
	Completion of Test Readiness Review	Task Lead
Process Controls	Management Reviews	Task Manager
	SQA process audits	SM
	Peer Reviews of Test procedures	Developer
	Code Walkthroughs	Developer
	Metrics, collection, analysis and reporting	Task Lead
Metrics	TBD	

4.3.4 BUILD AND RELEASE

Once the Task Lead's review is completed, the CM is notified that the code is ready for a CM build. The Build and Release step includes building, preserving, releasing and staging the software. This section only covers preparation and delivery of the software to internal sources for verification and validation purposes. Software released to a third party is covered under the heading of Software Delivery Section 4.3.7.

The following functions are required during the Build and Release step. The responsible party for each function is shown in **bold** type.

Print out STDR/SPR for verification - **CM** prints out all related STDRs/SPRs which have been approved for incorporation into the CM build.

Verify each approved software "item" is related to an approved STDR/SPR - **CM** verifies all approved items are exported to the proper CM work set and related to one of the approved STDRs/SPRs. **CM** then actions all approved STDRs/SPRs and the related items to the "Build_Release" state.

Initiate CM Build - **CM** incorporates all software items at "Build_Release" into the CM build trees on HP and Linux, as appropriate, and initiates the build process. Upon completion of the build process, **CM** checks the build log for errors to verify that the build was successful. If errors are found, the Release Coordinator is notified.

Preserve all executables and shared libraries - **CM** will initiate a process to preserve all executables and shared libraries from the successful build into PVCS and will verify that there were no errors reported in the preserve log.

Create Baselines - **CM** will create baselines containing all project deliverables and source code and will confirm that the baselines were created successfully.

Create Release from successful baseline - **CM** will create a release of the software deliverables contained in the baseline. **CM** will verify that all software deliverables have been placed in their proper runtime directory structure.

Stage release output - **CM** will 'stage' the release output by packaging it into a tar file(s) in preparation for delivery and installation.

Email Task Leads when software release is ready - **CM,** once satisfied that the software has been successfully baselined, released and staged, will notify the Release Coordinator and Task Lead(s) that the software package is ready to be burned on a CD and/or installed on a test machine.

Action STDRs/SPRs and software items to next state - **CM** will action all STDRs/SPRs from the status of "Build_Release" to "Test" and all software items from the status of "Build_Release" to "Delivered."

Table 4.3.4-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Build and Release step along with the responsible parties.

Table 4.3.4-1. Build and Release Input, Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Updated software items at "approved" state and in appropriate work set	Task Lead and verified by CM
	All approved items are related to approved STDRs/SPRs	
Input	Approved SPRs/STDRs	Task Lead, Task Manager
Output	Executables and shared libraries are preserved in PVCS.	CM
	Baseline is created in PVCS	CM
	Deliverables are released to the appropriate release directories	CM
	Email notification sent to Release Coordinator and Task Lead(s)	CM
	New tar and install files have been created in staging area	CM
	STDRs/SPRs at "Test" state	CM
	Updated software items at "Delivered" state	CM
Exit Criteria	Log files without errors from each step	CM
	STDR/SPR printouts given to the Release Coordinator and Task Lead(s)	CM
Process Controls	SQA process audits	SM
	Metrics, collection, analysis and reporting	Task Lead
Metrics	TBD	

4.3.5 SOFTWARE INTEGRATION TESTING (SwIT)

SwIT takes place in the development environment. Development Test Beds have been identified as NHD# machines and have the latest release installed. During this test cycle newer builds of the same version number may be installed to include new code or correct defects. Each project will define its own release schedule for testing.

Based on the Test Plan, the Test Coordinator assigns each change document (e.g., STDR, SPR) along with appropriate test procedures to the appropriate Testers. Change documents may represent tasks for new software enhancement or fixes for software defects. The Testers executes all the test procedures associated with the change documents along with any other related test procedures identified by developers or Task Leads. To pass a STDR for a major enhancement, the tester must determine that the basic required functionality works to an acceptable extent. This will most likely be a subjective decision by the tester based on the degree to which the basic core requirements are met. If the basic functionality is present, then new SPRs should be written by the tester identifying individual software defects which are found. If one or more major components of an enhancement do not work as required, then the STDR should be failed and returned to the developer for further modification. SPRs associated with software defects which do not pass are also returned to the developer.

The Test Coordinator records all the results into the Test Log and reports the results to the Task Lead. Task Lead then either reassigns the change documents to the developers or conducts a Integration Readiness Review (TRR) for the application.

The following functions are required during the SwIT step. The responsible party for each function is shown in **bold** type.

Create Test Plan - The **Task Lead and Test Coordinator** will create the Test Plan that describes the testing to be performed during SwIT. The Test Plan includes a test schedule and assigns test procedures and SPRs to the available developers for testing. See the MDL Standard, Guidelines and Procedures (NWS 2003b) for guidelines on how to prepare a test plan.

Test Plan Peer Review - The **Task Lead, Release Coordinator and Test Coordinator** should conduct a peer review of the test plan.

Create CD from staging - The **Release Coordinator** will create the CD from the stagging area to install on NHD# machines after they are informed by CM that the stagging is ready.

Verify test environment and AWIPS configurations - The **System Administrators** will install the AWIPS configuration baseline for testing machines that the Task Manager indicates is necessary for testing. The **Release Coordinator** reviews the test environment and AWIPS configuration.

Test Install and Uninstall scripts - The **Release Manager** will run and test the install and uninstall scripts during the installation cycle.

Execute Test Plan/Run Test Procedures - The **Developers** run the assigned test procedures for the STDRs/SPRs and record results. New SPRs are created for all failed test procedures.

Assign newly created SPRs to appropriate release. - The **Task Manager** will review, approve or reject, reassign or divert to another release all pending SPRs.

Prepare Test Log - The **Test Coordinator** prepares a test log showing the testing performed and the results of that testing. See the MDL Standards Guidelines and Procedures (NWS 2003b) for more information on how to prepare a test log.

Prepare User Guide - The **Documentation Lead** prepares a first draft of the User Documentation.

Prepare Release Note - The **Documentation Lead** prepares the first draft of the Release Notes.

Prepare Installation Instructions - The **Release Coordinator** updates the installation instructions based on testing.

Prepare Training Instructions - The **Documentation Lead** prepares training instructions to assist in training users on new or enhanced functionality.

Prepare NCF Checklist - The **Documentation Lead** prepares an NCF checklist to for the NCF support folks to assist in providing 24/7 support of the application.

Complete DAR Checklist - The **Task Lead** responds to the appropriate items in the DAR Checklist.

Complete Partial DAR material - The **Task Lead** prepares partial DAR material. Information is provided to SEC via email. A conference call will be conducted if needed to resolves questions/issues.

Conduct an Integration Readiness Review (IRR) - The **Task Manager** will conduct a review of the SwIT activities and determine the readiness for System Integration Testing (SIT). Participants should include Project Manager, Task Manager, System Engineer, Software Manager, Configuration Management, and DDT team. Action items (if necessary) are documented.

Table 4.3.5-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this SwIT step along with the responsible parties.

Table 4.3.5-1. SwIT Input and Output, Entrance and Exit Criteria, Process Controls and Metrics

Title	Description	Responsibility
Entrance Criteria	STDR/SPR printout for testing	CM
	CM informs Release Coordinator that staging is ready	CM
Input	New tar and install files	CM
	Test Procedures	Developer
Output	Test Plan	Test Coordinator
	Updated Test Procedures	Developers or Testers
	Completed Test Log	Test Coordinator or Task Lead
	SPRs assigned to a release	Task Manager
	Draft User Guide	Documentation Lead
	Draft Release Notes	Documentation Lead
	Draft Installation Instructions	Documentation Lead
	Draft Training Instructions	Documentation Lead
	DAR Checklist	Task Lead
	Partial DAR Material	Task Lead
	NCF Checklist	Documentation Lead
Exit Criteria	CD with the latest release/version with install and uninstall scripts	Release Coordinator
	Test log is completed, all “Fail” test procedures have an SPR # assigned to it	Test Coordinator or task Lead
	New SRS assigned to a release	Task Manager
	Conduct an Integration Readiness Review	Task Manager
Process Control	SQA process audits	SM
	Management Reviews	Task Manager
	Metrics, collection, analysis and reporting	Task Lead
	Formal review	Task Lead
Metrics	Defect Analysis	SM, Test Coordinator

4.3.6 SYSTEM INTEGRATION TESTING

SIT takes place in test environment. Test beds for this testing cycle have been identified as NMTW or NHDA machines. This testing takes place after the code freeze has been implemented and the latest builds from the development environment have been updated. SIT for some projects may be outsourced to NGIT for a complete or limited test cycle.

If errors are found in the application during this testing process, the Tester prepares a SPR to document the error. SPR is provided to Task Lead for resolution. The Task Lead assigns the SPRs either back to the developers to correct the error for the current release or assigns a future release to the SPR for resolution. After the application has been modified to address the SPRS, the updated code is resubmitted to CM, new media is cut and loaded back onto the NHDW platform, once the change is passed and accepted the same media is used to install the application on NMTW for the Testers to resume testing. This process is repeated until all the test procedures and change documents have been successfully executed.

The results of all formal application testing is provided to the Task Lead. This report contains the test procedures, test data, and test results, including a history of failed test cases. Copy of the completed Test Plan is attached in the Software Development Files (SDFs).

The following functions are required during the SIT step. The responsible party for each function is shown in **bold** type.

Create CD from staging - The **Release Coordinator** will create the CD from the stagging area to install on NHDA or NMTW machines after they are informed by CM that the stagging is ready.

Test Install and Uninstall scripts - The **Release Manager** will run and test the install and uninstall scripts during the installation cycle.

Create Test Plan - The **Task Lead and Test Coordinator** will create the Test Plan that describes the testing to be performed during SIT. The Test Plan includes a test schedule and assigns test procedures and SPRs to the available developers for testing.

Run Test Procedures - The **Developers** run the assigned test procedures for the STDRs/SPRs and record results. New SPRs are created for all failed test procedures.

Assign newly created SPRs to appropriate release. - The **Task Manager** will review, approve or reject, reassign or divert to another release all pending SPRs.

Prepare Test Log - The **Test Coordinator** prepares a test log showing the testing performed and the results of that testing.

Prepare Final User Guide - The **Documentation Lead** prepares a final of the User Documentation.

Prepare Final Release Note - The **Documentation Lead** prepares the final of the Release Notes.

Update Installation Instructions - The **Documentation Lead** updates the installation instructions based on testing.

Prepare Final Training Instructions - The **Documentation Lead** updates the training instructions.

Table 4.3.6-1 describes the input, output, entrance and exit criteria, process control, and metrics for this SIT step along with the responsible parties.

Table 4.3.6-1. System Integration Testing Input, Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	STDR/SPR printout for testing	Task Lead or Test Coordinator
	All STDRs/SPRs are closed	Task Manager
	New tar and install files are ready in staging area for CD creation	CM
Input	Master CD based on latest release	CM and Release Coordinator
	Test Plan	Test Coordinator
	Updated Test Procedures	Test Coordinator
	Draft User Guide	Documentation Lead
	Draft Release Notes	Documentation Lead
	Draft Installation Instructions	Release Coordinator
	Draft Training Documentation	Documentation Lead
Output	Test Log	Test Coordinator
	Master CD	Release Coordinator
	Installation Instructions	Release Coordinator
	Release Notes	Documentation Lead
	User Guide	Documentation Lead
	Modification Note	Documentation Lead
	Version description Document	Documentation Lead
	Training Documentation	Documentation Lead
Exit Criteria	CD with the latest build/version with install and uninstall scripts	Task Lead
	All STDRs/SPRs are tested and results are recorded on the log	Developers or Testers
	Test Log is complete, all "Fail" test procedures have a SPR # assigned to it	Test Coordinator
Process Control	SQA process audits	SM
	Management Reviews	Task Manager
	Metrics, collection, analysis and reporting	Task Lead
Metrics	Defect Analysis	Test Coordinator

4.3.7 SOFTWARE DELIVERY

This section defines the MDL software delivery of AWIPS applications to third party for System Acceptance Testing and distribution.

AWIPS is in a phase where both development and sustaining engineering of current capabilities will take place. AWIPS has followed, and will continue to follow, a release based development process, grouping changes into larger releases that go through a controlled development and testing process. Three different types of releases are expected:

Major Releases - involve highly complex changes that typically extend or change the underlying system infrastructure and impact many applications. This software is tested by NGIT and deployed by OOS and NGIT.

Maintenance Releases - are much more restricted in scope and typically apply to groups of independent applications. This software is tested and deployed by SEC.

Emergency Releases - are issued in response to critical discrepancy reports which significantly compromise the user's ability to perform their missions. This software is tested by MDL and deployed by SEC.

Alpha Testing Releases - are issued to allow testing of critical functionality at a field site. The software is tested and deployed by MDL.

A quick response cycle of development is also established to accommodate some baseline or maintenance release as needed. This is called a *RAP* cycle for Rapid Alpha Process. RAP cycles are typically six to eight weeks long from development to testing at Alpha sites. Following are the paths for Standard and RAP development cycles:

Standard cycle - The developers will perform development and unit testing. Integration testing will be performed in-house on the NHDx and NHDA systems. Hand-off to NGIT is done for System Integration Testing (SIT) and System Acceptance Testing (SyAT). NGIT will perform baseline testing on preselected sites. After successful testing on baseline sites, the software is ready for general release.

RAP cycle - The developers will perform development and Unit testing. Integration testing will be performed in-house on NHD#s machines. LSIT will be performed before the software is released for ALPHA testing to the preselected sites. After ALPHA testing has reported its results and all corrections are completed the software is released for LSyAT. Once the software has passed LSyAT, it's ready for general release.

The delivery of AWIPS software includes the following:

Major Releases - is delivered to NGIT and includes:

- HP source code
- Linux deliverables as tar files
- List of TDRS/STDRs/SPRs covered
- Installation Instructions
- Release Notes
- NCF Checklist
- User Guide
- Test Procedures
- Design Approach Review (DAR) Questionnaire
- Modification Note

- Version Description Document
- Training Document

Maintenance Releases - is delivered to SEC and includes:
- Either executables or source code, as required
- Installation scripts and instructions
- Modification Note

Emergency Releases - same as Maintenance Release

Alpha Testing and Baseline Releases - is delivered to NGIT and includes:
- Master CD or post files on NOAA server depending on size
  Associated documentation
- Installation Instructions
- Release Notes
- User Guide
- Modification Note
- Training Document

The Major Release, Maintenance Release and Emergency Release delivery follows the Standard Cycle of Development, where as Alpha Testing Release delivery primarily follows the RAP Cycle of Development.

5.0 DOCUMENTATION

Documentation is continually prepared to communicate and archive valuable development information. This information is used by management, system integrators, users, developers and support and maintenance personnel. It is included as a separate section in this document to emphasize its importance.

5.1 FORMAL DOCUMENTATION

Documentation is prepared in accordance with the guidelines provided in the MDL Standards, Guidelines and Procedures (NWS 2003b). Documentation will be updated as appropriate. Application documentation is updated for each release of the software. Development documentation is updated continuously and archived in each application Software Development Files (SDFs). Management and project tracking information is kept in the Project Management Files (PMFs).

This section identifies what documents are produced during the development life cycle. See the MDL Standards, Guidelines and Procedures (NWS 2003b) and the examples section of the MDL Development Library for more information concerning the following documentation.

Task Development Report (TDRs) - Formal configuration item used to track AWIPS development tasks. These tasks are commonly included in the MDL Staffing Plan and are assigned to a Project or Task Lead.

Sub-task Development Report (STDR) - Formal configuration item used to assign development tasks to individual developers.

Software Problem Report (SPRs) - Formal configuration item to track software deficiencies and are assigned to a specific release and developer.

Project Tracking Information (PTI)

- Development Schedule - High level schedule showing the development activities (e.g., requirement review) and major checkpoints on a time line.
- Staffing Plan - List of tasks (TDRs/STDRs) per release with estimated level of effort and identified Task Manager, Task Lead and development staff.
- Development Checklist - Internal detailed checklist employed by the management to track development progress and adherence to the MDL Software Development Approach requirements (NWS 2004a).

Requirements Description (RD) - Formal documentation of the requirements agreed to with OCWWS. These requirements are trackable to all aspects of the development cycle.

Design Document - Documentation used by MDL to capture the design of an application. Documentation may include site-level diagrams, Data Flow Diagrams (DFDs).

Test Procedures - For a given unit, module, or application, one or more test procedures are identified to evaluate the functional or structural condition of the code. Test procedures are designed based on specific functional requirements or components of code structure. Each test procedure will identify the software requirements validated by the test.

Installation Instructions - Instructions used to install MDL application software. The Installation Instructions usually includes the following sections:

- Prerequisites
- Pre-install Preparations
- "The" Install Instructions
- Post-Install Instructions
- Deinstall Instructions
- Attachments
  - New/Merge/Replace File List
  - Install Space Requirements
  - Sample Install/Deinstall Logs
  - Customer Support Team (CST) Contact Information

User Documentation - Web and hard copy documentation prepared to instruct users how to use and configure application software. The User Documentation usually contains documentation on all functionality, but for each specific release, it features and highlights what's new in the release. The new or updated functionality usually takes the form of one or more of the following:

- Overview
- Technical
- Customization
- Troubleshooting
- FAQ

Release Notes - Release notes document new functionality, known software deficiencies, fixes to software deficiencies. Release Notes usually contain the following 4 different sections:

- New Functionality (Describes, at a high level, any new functionality introduced in the release)
- News You Can Use (Describes any workarounds and/or lessons learned)
- Problems Fixed (Describes FRs/SPRs fixed in the current release)
- Problems to be Fixed in a Future Release (Describes all SPRs targeted for a future release)

Modification Notice - A “Mod Note” provides authorization for sites to add/modify software to their CM AWIPS systems.

Training Documentation - Documentation used by OCWWS to train forecasters in new or enhanced techniques

DAR Checklist - The Design Approach Review (DAR) Checklist includes:

- Requirement Assessment
- Design/Architecture Assessment
- Operability
- Application Integration and Installation
- System Integration
- Performance
- Hardware/Resource Usage Design
- External Interfaces
- General design Assessment
- Responsible Individuals
- Testing Assessment
- Configuration management

- Schedule

Partial DAR material - Material to be delivered to NGIT for System Integration Testing. This material includes:
- brief description of functionality
- special conditions, environment set-up, data needed for testing
- impacts to AWIPS Common, COTS, or other critical interfaces identified by SEC.

Table 5.1-1 summarizes the procedures and responsibilities for producing/updating, reviewing, and approving. It also indicates which documents are under configuration management.

Table 5.1-1. Procedures and Responsibilities for Producing/Updating, Reviewing, Approving, Controlling Documentation

Document	**Preparer**	**Approving Power**	**Under CM**
TDR	Task Manager	Project Manager	Yes
SPR	Task Manager	Project Manager	Yes
Project Tracking Information (PTI)	Task Manager	Project Manager	Yes
Requirements Description	Task Lead	Task Manager	Yes
Design Documentation	Design Lead	Task Manager	Yes
STDR	Design Lead	Task Manager	Yes
Test Procedures	Developers	Task Lead	Yes
Installation Instructions	Task Lead and Release Coordinator	Task Manager	Yes
User Documentation	Documentation Lead	Task Manager	Yes
Release Notes	Documentation Lead	Task Manager	Yes
DAR Checklist	Task Lead	Task Manager	Yes
Partial DAR Material	Task Lead	Task Manager	Yes
Modification Note	Documentation Lead	Task Manager	Yes
Training Documentation	Documentation Lead	Task Manager	Yes

5.2 INFORMAL DOCUMENTATION

A software project's informal documentation consists of a set of Software Development Files (SDFs). The SDF contains information describing the development or maintenance of the software product. The SDF should be maintained online as much as possible, to facilitate searching and inclusion into documentation. It may contain media copies or references to controlled media copies of supporting data. As a minimum, each SDF will contain the following:

- formal documentation and a master document list,
- Briefing slides,
- Design information and documentation, including rationale supporting design decisions,
- Peer review, design review, and code walkthrough results, checklists, and comments,
- Test Plan and Test Logs
- All engineering and formal CM change history,
- Meeting minutes, memos, action items, checklists, and correspondences, and
- A log of technical lessons learned that could be of value in the future to this product or other efforts.

6.0 REVIEWS

The purpose of this section is to identify the types of reviews employed and when they occur. The guidance addressing review frequency, preparation, participant number and characteristics, formality, and recording and closeout of issues can be found in the MDL Standards, Guidelines and Procedures (NWS 2003b) document.

This section defines the process for the five types of internal reviews that are performed during the development of software. These reviews are categorized as:

- Peer reviews
- Code Walkthroughs
- Project Status Reviews
- Development Reviews

Some of these reviews are conducted internal to MDL and, in most cases, do not include customer participation. Typically, the reviews addressed in this section lead up to development reviews (e.g., design reviews) and formal interaction with the customer.

Our experience has proven the effectiveness of internal reviews throughout the software development and maintenance process. This is an extremely cost-effective approach for early identification and resolution of technical and management problems and improved communication within the software project team. Furthermore, the types of reviews defined in this section work equally well on all sizes and types of software projects. However, each type of review must be exercised in an appropriate manner, as defined in this section, or substantial benefits may be degraded.

6.1 PEER REVIEWS

The concept behind peer reviews is that the author/developer of a product (i.e., specification, design, unit of code, test procedures) gets help from a colleague(s) who is familiar with the product. Together, they discuss in detail a specific portion of the overall product. The author presents the product element to the colleague(s), item by item, who in turn raises questions and suggestions. Application of the concept is simple and inexpensive. Peer reviews are ad hoc. They are accomplished with only enough planning necessary to solicit participation from the colleague(s) and prepare the product element to a state where it can be reviewed. Such reviews should always be limited to two hours. To accomplish that peer reviews usually examine a portion of a product rather than the entire document, plan, specification, or design under review. Because peer reviews impose only small blocks of time, this technique is used frequently among the set of people working that project. Notes, issues and action items are kept by the author of the product element. No formal "list of issues" or action items result from one-on-one reviews. The list of issues packaged with a minimal amount of data about the review itself (e.g., date and members) are placed in the Software Development Files (SDFs). Management personnel track these reviews and ensure they are occurring through project status reviews.

The MDL Standards, Guidelines, and Procedures (NWS 2003b) contains guidelines for conducting Peer Reviews.

6.2 CODE WALKTHROUGHS

Similar to peer reviews and code walkthroughs involve only technical staff. The number of participants, counting the product author, ranges from three to six. Scheduled peer reviews have a maximum duration of two hours. The focus is on identifying technical issues and concerns, not solutions. As discussed below, a moderator is assigned to keep the review focused only on technical issues, rather than discussing solutions to issues.

Follow-up meetings are conducted as appropriate to the importance of the identified issues. All code walkthrough documentation are placed in the Software Development Files (SDFs) for the product that was partially or fully reviewed.

The MDL Standards, Guidelines and Procedures (NWS 2003b) contains a set of sample checklists that can be used to ensure a consistent and relatively complete review of various software products, and a sample form that can be used to record and track comments and issues generated during scheduled peer reviews.

6.3 MANAGEMENT REVIEWS

Management is responsible for resolving the technical, schedule, and resource issues that are a significant risk to the project. Primary communication and resolution mechanisms used by management are Project Status Reviews.

6.3.1 PROJECT STATUS REVIEWS

Project Status Reviews occur both on a periodic and event-driven basis. Participants are the leads for the various elements of the project, for instance, Task Manager, Task Lead, DDT team members, System Administration (SA), Software Manager (SM) and Configuration Management (CM) always should be present. Technical progress, plans, performance, and issues are discussed and tracked against the baselined project plans. Each attendee presents a summary of activities and issues since the last Project Status Review as well as plans for the upcoming period. The meeting focuses on open, significant issues. Anyone can present alternative solutions for those significant issues.

Task Manager should record all issues identified at the meeting as requiring resolution. An action list is distributed by the **Task Manager** to the meeting attendees. The **Task Manager** maintains the list and detailed records of how each issue was resolved.

6.3.2 DEVELOPMENT REVIEWS

Development reviews are conducted upon the completion of a formal milestone. A key purpose of the review is to present the accomplishments and results of the software project up to the current milestone with other NWS organizations. In many cases approval may be required to proceed to the next step. Review material focuses on the key points of the project activities and project status. Activities performed since the last review, planned activities, and the status of all open items identified in previous reviews are discussed. In addition a standard element of all preparatory reviews is explicit attention to project risks including schedule, resources, and technical. Formal reviews include:

Requirements Review - Presentation and request for approval of requirements. See the MDL Standards, Guidelines and Procedures (NWS 2003b) for requirement review guidelines

Design Review - Presentation and request for approval of design information. See the MDL Standards, Guidelines and Procedures (NWS 2003b) for design review guidelines. Note: For the AWIPS project, a updated DAR Checklist should be presented at the review.

Test Readiness Review (TRR) - The purpose of the Test Readiness Review (TRR) is to evaluate the software, development process, and test procedure developed during the informal application testing to certify that all software requirements are properly validated. The TRR is performed by the Task Lead(s), Release Coordinator, the developers, and Software Manager and indicates readiness for Software Integration Testing (SwIT). Evaluation process involves reviewing the test procedures and available test data for each test procedure. Requirement, Design, and Use Case documents along with code walkthrough results are used as supporting information during the evaluation process. Upon approval, the test procedures are submitted for formal application testing.

The TRR includes:

- Verify that all development steps were completed
- Verify documentation is complete
  - Installation Instructions
  - Test Procedures
- Verify integration material is complete
  - DAR Checklist
  - Partial DAR material
- Verify building and release information
- Coordinate build schedule with CM
- Verify test environment and AWIPS configurations are ready on the test bed

Integration Readiness Review (IRR) - The purpose of the Integration Readiness Review (IRR) is to evaluate the application's readiness to proceed to an integration platform for System Integration Testing. The IRR is performed by the Task Lead(s), Release Coordinator, the developers, and Software Manager. Evaluation process involves reviewing the test results of Software Integration Testing and to ensure all defects are either corrected and tested or have been reassigned to a later release. A satisfactory resolutions of all change documents along with any open SPRs is essential before approval.

The IRR includes:

- Verify software was adequately testing
  - Review of the Test Plan
  - Review of test logs
  - Review and document outstanding problems
- Verify documentation is complete
  - User Guide
  - Release Note
  - Installation Instructions
  - Training Instructions
  - NCF Checklist
- Verify integration material is complete
  - DAR Checklist
  - Partial DAR material

7.0 TESTING

The primary goal of all of the testing activities is to identify and remove defects and to provide a standard approach for testing the MDL software applications.

The MDL Test program is based on the following key concepts:

- The testing approach is to provide an effective, repeatable, software test process which is independent of the software language, design methodology, and development environment,
- The testing scope is to identify and remove all defects and also to validate that software applications meet all requirements allocated to them,
- The testing strategy incorporates two basic points of view: functional (user's) and structural (program attributes). The testing of each application will be designed to include adequate coverage of both the functional and structural aspects,
- The testing process will essentially follow a bottom-up approach. The testing will begin at the lowest unit level and proceed upward as units are integrated into the application. Regression testing will be performed when appropriate, and
- The applications will be handed over to NGIT for the final integration into the AWIPS system, and
- Quality is designed into products using defined processes that are continually monitored and updated to improve their efficiency, to avoid recurring problems, and to maintain the desired quality of resulting products.

These concepts are accomplished by implementing the following:

- Informal testing,
- Test Readiness Review,
- Formal testing,
- Configuration management,
- Non-conformance reporting and corrective action, and
- Training.

The MDL Standards, Guidelines and Procedures (NWS 2003b) describes the criteria, responsibilities, and test strategy (i.e., test cases, procedures, level of testing) necessary to provide an effective, repeatable software test process which is independent of the test environment.

8.0 CONFIGURATION MANAGEMENT

The MDL Configuration Management (CM) function ensures that the software development process is followed and that the necessary metrics are collected.

The MDL CM program is based on the following key concepts:

- The tools used (PVCS Dimensions for AWIPS) are customized to the defined development life cycle,
- All code changes are documented and related to the appropriate change document, and
- The change documents are customized to collect the necessary metrics and to provide management, developers, and users with the appropriate information in a timely manner, so as to identify risks as early as possible.

These concepts are accomplished by implementing the following:

- Creating the necessary development and build environments,
- Assigning the correct roles for the development/test staff,
- Updating the change documents to support the project,
- Establishing the daily development builds,
- Creating the test releases, and
- Creating the baselines for alpha test and full deployment.

The Configuration Management Plan for AWIPS (NWS 2003a) provides the detailed configuration management procedures applicable to deliverables: code, associated data files and documentation, responsibilities, tools and techniques used, the stage at which items are brought under configuration control and change control management.

9.0 SOFTWARE STANDARDS

The critical importance of developing well documented and well-structured code has become more obvious with time. Except for, possibly, some small programs/subroutines written exclusively to test an idea or structure that will soon be discarded, Government developed software will be inherited and maintained by others. It is imperative to follow good coding and documentation rules in the development of all code, and in particular code that is to be handed off for use outside of MDL. Reasons include:

- With several people involved in a project, it is important that guidelines be followed so that all can easily "read" another person's program,
- Usually, it will fall to someone other than the originator to modify or maintain a program at some time in the future. Again, if a program has been written and documented according to prescribed rules, revisions and maintenance are much easier,
- Code developed by the DDTs is for the express purpose of implementation and integration into a much larger system. If all such code follows the same guidelines, understanding and dealing with it will be much easier, and we will be able to answer questions more readily than otherwise,
- Standardization will reduce errors in coding. The eye and mind become accustomed to "patterns," and a break in a pattern may be an error,
- Converting a body of software from one computer system to another is easier if it is all written and documented to the same standards, and
- New employees with little or no programming experience can be more easily trained in good procedures if those procedures are written down and everyone follows them.

In summary, the objectives of these guidelines are to enhance clarity, testability, maintainability, and person-to-person and computer-to-computer transferability of software throughout its life cycle. The following standards are contained in the MDL Standards, Guidelines and Procedures (NWS 2003b):

- Fortran,
- C,
- C++,
- Tcl/Tk,
- Python,
- scripts, and
- HTML.

10.0 ENVIRONMENTS

10.1 DEVELOPMENT

OST software development, software compilation and builds are performed on the following two suites of Software Development and Build systems. These systems are NOT on the AWIPS WAN. They have address space on the SSMC2 Building LAN, and they are protected by OST managed firewalls. The NHD (National Headquarters Development) Suite of Software Development and Build support MDL software development. This suite of systems is located in the 7th floor computer area in SSMC2 and in the MDL computer areas on the 10th floor of SSMC2.

The development environment for MDL consists of legacy HP, LINUX workstations and PC's running a variety of X app software applications. The following HP equipment, broken out by floor, is used for development:

7th floor - ds1-nhdw, fs1-nhdw, bd1-nhdw, bd2-nhdw, bd3-nhdw, ws1-nhdw, ws2-nhdw, ws3-nhdw, ws6-nhdw, ws7-nhdw

10th floor - w10, w11, w12, w13, w14, w15, w16, w18

The following LINUX equipment, broken out by floor, is used for development:

7th floor - LX1-nhdw, LX3-nhdw, LX4-nhdw, LX6-nhdw, LX7-nhdw, LX8-nhdw, TDL3, IFPS

10th floor - LX2-nhdw

The MDL environment is divided into the developers, workspace and the project workspace. The coding, development, and testing of units and modules will be conducted in the developers' workspace under control of the developers. The testing of the integrated modules which comprise the application will be conducted in the project workspace.

10.2 TESTING

MDL employs a variety to test environments and tools to support application testing prior to release to a third party.

The target hardware and interfaces for the system should be used to the greatest extent possible. However, test tools should be used to control the test process, inject test data, facilitate data capture, and test in an environment where the full system is not available.

Two environments will be available for testing the MDL applications software, one for the Software Integration Testing (SwIT) and one for System Integration Testing (SIT). The System Administrators are responsible for maintaining the hardware and COTS software of that system. The **Test Coordinator** is responsible for maintaining the appropriate AWIPS software on the appropriate test bed.

The SwIT environment consists of an NHD2, NHD3, NHD4, and NHD5 each consisting of one or two Linux workstations, one HP workstation, and one HP DS/AS (combined). Multiple platforms (4) are required because of the overlapping release structure of AWIPS deliveries.

The SIT environment consists of the NHDA and NMTW. The NHDA is a combined AWIPS WFO and AWIPS RFC system with localization setup for Sterling Virginia, and a secondary localization of Albany NY (ALY). In addition, NHDA has an RFC database and Radars set up like the Arkansas/Red Basin RFC (ABRFC) in Tulsa

Oklahoma to support software testing for Office of Hydrologic Development (OHD). The National Headquarters Modernization Test and Integration System (NMT) located at NWS Headquarters in Silver Spring, Maryland. The NMT will be separate and independent of the NHDA. The NMT will be configured to represent a joint WFO and River Forecast Center (RFC) operational environment. An additional SIT site is the AWIPS Test Bed located at NGIT in McLean, Virginia.

12.0 REFERENCES

CSC, 2000: National Weather Service Advanced Weather Interactive Processing System Software Development Plan. Contract Number 263-96-D-0322, Task Number DOC-NOAA-1998-C-1523, Doc. ID No. AWIPS SDP-01-00, CSC, Rockville, Maryland.

National Weather Service, 1995a: Software Development Plan for Producing WFO Hydrometeorological Applications for AWIPS, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce.

National Weather Service, 1995b: Test Plan for Producing WFO Hydrometeorological Applications for AWIPS, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce.

National Weather Service, 2003a: MDL Configuration Management Plan for AWIPS, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S Department of Commerce, (in preparation).

National Weather Service, 2003b: MDL Standards, Guidelines and Procedures, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, (in preparation).

National Weather Service, 2004a: MDL Software Development Approach, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, (draft).

Installation & Testing Process on the AWIPS Test Systems: NMTW, NMTR, and NHDA, NWS 2002e: Systems Engineering Center, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, 22p.